Контрольная работа №1

Универсальные типы. Классы-коллекции. Методы расширения класса System.Linq.Enumerable

Цель: овладение основными приемами работы с классами-коллекциями.

Задачи:

1) Изучить основные понятия лекции на тему «Интерфейсы. Коллекции».
2) Выполнить задания (см. раздел Задания) в соответствии с вариантом.

Задания

Общая часть (для варианта 1 и варианта 2)

Во всех вариантах требуется определить универсальный делегат *delegate System.Collections.Generic.KeyValuePair<TKey,TValue> GenerateElement<TKey, TValue>(int j);*
и универсальный класс *TestCollections<TKey,TValue>* (*TestCollections<TKey>*), который содержит закрытые поля следующих типов

- System.Collections.Generic.List<TKey>;
- System.Collections.Generic.List<TValue>;
- System.Collections.Generic.Dictionary<TKey, TValue>;
- System.Collections.Generic.Dictionary<string, TValue>;
- GenerateElement<TKey, TValue>.

Конкретные значения типовых параметров TKey и TValue зависят от варианта.

В конструкторе класса *TestCollections<TKey,TValue>* создаются коллекции с заданным числом элементов. Надо сравнить время поиска элемента в коллекциях-списках List<TKey>, время поиска элемента по ключу и элемента по значению в коллекциях-словарях Dictionary<TKey,TValue>.

Для автоматической генерации элементов коллекций надо определить метод, который принимает один целочисленный параметр типа *int* и

возвращает ссылку на объект типа *KeyValuePair<TKey,TValue>*. Метод должен инициализировать объекты *KeyValuePair<TKey,TValue>* так, чтобы соответствие между номером элемента и объектом *TKey* в паре ключ-значение было взаимно-однозначным.

Метод для автоматической генерации элементов коллекций передается в класс *TestCollections<TKey,TValue>* через параметр конструктора класса. Для этого в классе *TestCollections<TKey,TValue>* надо определить конструктор с двумя параметрами, имеющими тип *int* и *GenerateElement*. Через целочисленный параметр объектам класса передается число элементов в коллекциях, через экземпляр делегата *GenerateElement* – метод, который используется для автоматической генерации пары ключ-значение в виде объекта *KeyValuePair<TKey,TValue>*.

Число элементов в коллекциях пользователь вводит в процессе работы приложения. Если при вводе была допущена ошибка, приложение должно обработать исключение, сообщить об ошибке ввода и повторить прием ввода до тех пор, пока не будет правильно введено целочисленное значение.

Для четырех разных элементов – первого, центрального, последнего и элемента, не входящего в коллекцию, – надо измерить время поиска

- элемента в коллекциях *List<TKey>* и *List<string>* с помощью метода Contains;
- элемента по ключу в коллекциях *Dictionary<TKey, TValue>* и *Dictionary<string, TValue>* с помощью метода *ContainsKey*;
- значения элемента в коллекции *Dictionary<TKey, TValue>* с помощью метода *ContainsValue*.

Так как статический метод для автоматической генерации элементов должен обеспечивать взаимно-однозначное соответствие между значением целочисленного параметра метода и объектами *TKey*, его можно использовать как при создании коллекций, так и для генерации элемента для поиска.

Вариант 1

Определить класс ***Exam***, имеющий

- свойство типа *string*, в котором хранится название предмета;
- свойство типа *int*, в котором хранится оценка;
- свойство типа *System.DateTime* для даты экзамена.

Определить класс ***Test***, имеющий

- свойство типа *string*, в котором хранится название предмета;
- свойство типа *string*, в котором хранится отметка о сдаче зачета;
- свойство типа *System.DateTime* для даты зачета.

В классе ***Exam*** определить:

- конструктор с параметрами типа *string*, *int* и *DateTime* для инициализации всех свойств класса;
- конструктор без параметров, инициализирующий все свойства класса некоторыми значениями по умолчанию;
- перегруженную(override) версию виртуального метода *string ToString()* для формирования строки со значениями всех свойств класса.

Определить класс ***Student***, имеющий

- закрытое поле типа *Person* (описание класса разработать самостоятельно), в котором хранятся данные студента;
- закрытое поле типа *Education* (описание класса разработать самостоятельно) для информации о форме обучения;
- закрытое поле типа *int* для номера группы;
- закрытое поле типа *Exam* [] для информации об экзаменах, которые сдал студент.

В классе ***Student*** определить конструкторы:

- конструктор с параметрами типа *Person*, *Education*, *int* для инициализации соответствующих полей класса;
- конструктор без параметров, инициализирующий поля класса значениями по умолчанию.

В классе ***Student*** определить

- свойство типа ***Person*** для доступа к полю с данными студента;
- свойство типа ***Education*** для доступа к полю с формой обучения;
- свойство типа ***int*** для доступа к полю с номером группы;
- свойство типа ***Exam []*** для доступа к полю со списком экзаменов.

В классе ***Student*** определить

- свойство типа *double* (только с методом *get*), в котором вычисляется средний балл как среднее значение оценок в списке сданных экзаменов;
- индексатор булевского типа (только с методом *get*) с одним параметром типа *Education*; значение индексатора равно *true*, если значение поля с формой обучения студента совпадает со значением индекса, и *false* в противном случае;
- метод *void AddExams (params Exam [])* для добавления элементов в список экзаменов;
- перегруженную версию виртуального метода *string ToString()* для формирования строки со значениями всех полей класса, включая список экзаменов;
- виртуальный метод *string ToShortString()*, который формирует строку со значениями всех полей класса без списка экзаменов, но со значением среднего балла.

В класс ***Exam*** добавить реализацию интерфейсов

- *System.IComparable* для сравнения объектов типа *Exam* по названию предмета;
- *System.Collections.Generic.IComparer<Exam>* для сравнения объектов типа *Exam* по оценке.

Определить вспомогательный класс, реализующий интерфейс *System.Collections.Generic.IComparer<Exam>*, который можно использовать для сравнения объектов типа *Exam* по дате экзамена.

В классе ***Student*** для списков зачетов и экзаменов использовать типы

- *System.Collections.Generic.List<Test>* для списка зачетов;

- *System.Collections.Generic.List<Exam>* для списка экзаменов.

В классе *Student* определить методы для сортировки списка экзаменов

- по названию предмета;
- по оценке / отметке о сдаче зачета;
- по дате зачета / экзамена.

Определить универсальный делегат

delegate TKey KeySelector<TKey>(Student st);

Определить универсальный класс ***StudentCollection<TKey>***, содержащий коллекцию объектов *Student*, в котором для хранения коллекции используется тип *System.Collections.Generic.Dictionary<TKey, Student>*. Типовой параметр *TKey* универсального класса *StudentCollection<TKey>* определяет тип ключа в коллекции *Dictionary<TKey, Student>*.

Метод, который используется для вычисления ключа при добавлении элемента Student в коллекцию класса *StudentCollection<TKey>*, отвечает делегату *KeySelector<TKey>* и передается *StudentCollection<TKey>* через параметр единственного конструктора класса.

Класс *StudentCollection<TKey>* содержит

- закрытое поле типа *System.Collections.Generic.Dictionary<TKey, Student>*;
- закрытое поле типа *KeySelector<TKey>* для хранения экземпляра делегата с методом, вычисляющим ключ для объекта *Student*;
- конструктор с одним параметром типа *KeySelector<TKey>*;
- метод *void AddDefaults()*, с помощью которого можно добавить некоторое число элементов типа *Student* для инициализации коллекции по умолчанию;
- метод *void AddStudents (params Student[])* для добавления элементов в коллекцию *Dictionary<TKey, Student>*;
- перегруженную версию виртуального метода *string ToString()* для формирования строки, содержащей информацию обо всех элементах

коллекции *Dictionary<TKey, Student>*, в том числе значения всех полей класса *Student*, включая список зачетов и экзаменов;

- метод *string ToShortString()*, который формирует строку с информацией обо всех элементах коллекции *Dictionary<TKey, Student>*, состоящую из значений всех полей, среднего балла, числа зачетов и экзаменов для каждого элемента *Student*, но без списка зачетов и экзаменов.

В классе ***StudentCollection<TKey>*** определить свойства и методы, выполняющие операции со словарем *Dictionary<TKey,Student>* с использованием методов расширения класса *System.Linq.Enumerable* и статические методы-селекторы, которые необходимы для выполнения соответствующих операций с коллекцией:

- свойство типа *double* (только с методом *get*), возвращающее максимальное значение среднего балла для элементов *Dictionary<TKey,Student>*; если в коллекции нет элементов, свойство возвращает некоторое значение по умолчанию; для поиска максимального значения среднего балла надо использовать метод *Max* класса *System.Linq.Enumerable*;
- метод *IEnumerable<KeyValuePair<TKey,Student>>EducationForm (Education value)*, возвращающий подмножество элементов коллекции *Dictionary<TKey,Student>* с заданной формой обучения; для формирования подмножества использовать метод *Where* класса *System.Linq.Enumerable*;
- свойство типа *IEnumerable<IGrouping<Education,KeyValuePair<TKey, Student>>>* (только с методом *get*), выполняющее группировку элементов коллекции *Dictionary<TKey, Student>* в зависимости от формы обучения студента с помощью метода *Group* класса *System.Linq.Enumerable*.

В методе ***Main()***

1. Создать объект типа *StudentCollection*. Добавить в коллекцию несколько разных элементов типа *Student* и вывести объект *StudentCollection*.

2. Для созданного объекта *StudentCollection* вызвать методы, выполняющие сортировку списка *List<Student>* по разным критериям, и после каждой сортировки вывести данные объекта. Выполнить сортировку

- по фамилии студента;
- по дате рождения;
- по среднему баллу.

3. Вызвать методы класса *StudentCollection*, выполняющие операции со списком *List<Student>*, и после каждой операции вывести результат операции. Выполнить

- вычисление максимального значения среднего балла для элементов списка;
- фильтрацию списка для отбора студентов с формой обучения *Education.Form*;
- группировку элементов списка по значению среднего балла; вывести все группы элементов.

4. Создать объект типа *TestCollection*. Вызвать метод для поиска в коллекциях первого, центрального, последнего и элемента, не входящего в коллекции. Вывести значения времени поиска для всех четырех случаев. Вывод должен содержать информацию о том, к какой коллекции и к какому элементу относится данное значение.

Вариант 2

Определить класс ***Article***, имеющий

- свойство типа *Person*, в котором хранятся данные автора статьи;
- свойство типа *string* для названия статьи;
- свойство типа *double* для рейтинга статьи.

В классе ***Article*** определить:

- конструктор с параметрами типа *Person, string, double* для инициализации всех свойств класса;
- конструктор без параметров, инициализирующий все свойства класса некоторыми значениями по умолчанию;
- перегруженную(override) версию виртуального метода *string ToString()* для формирования строки со значениями всех свойств класса.

Определить класс ***Magazine***, имеющий

- закрытое поле типа *string* с названием журнала;
- закрытое поле типа *Frequency* с информацией о периодичности выхода журнала;
- закрытое поле типа *DateTime* с датой выхода журнала;
- закрытое поле типа *int* с тиражом журнала;
- закрытое поле типа *Article []* со списком статей в журнале.

В классе ***Magazine*** определить конструкторы:

- конструктор с параметрами типа *string, Frequency, DateTime, int* для инициализации соответствующих полей класса;
- конструктор без параметров, инициализирующий поля класса значениями по умолчанию.

В классе ***Magazine*** определить

- свойство типа *string* для доступа к полю с названием журнала;
- свойство типа *Frequency* для доступа к полю с информацией о периодичности выхода журнала;
- свойство типа *DateTime* для доступа к полю с датой выхода журнала;

- свойство типа *int* для доступа к полю с тиражом журнала;
- свойство типа *Article []* для доступа к полю со списком статей.

В классе ***Magazine*** определить

- свойство типа *double* (только с методом *get*), в котором вычисляется среднее значение рейтинга в списке статей;
- индексатор булевского типа (только с методом *get*) с одним параметром типа *Frequency*; значение индексатора равно *true*, если значение поля типа *Frequency* совпадает со значением индекса, и *false* в противном случае;
- метод *void AddArticles (params Article[])* для добавления элементов в список статей в журнале;
- перегруженную версию виртуального метода *string ToString()* для формирования строки со значениями всех полей класса, включая список статей;
- виртуальный метод *string ToShortString()*, который формирует строку со значениями всех полей класса без списка статей, но со значением среднего рейтинга статей.

Определить класс ***Edition***. Класс ***Edition*** имеет

- защищенное(protected) поле типа *string* с названием издания;
- защищенное поле типа *DateTime* с датой выхода издания;
- защищенное поле типа *int* с тиражом издания.

В классе ***Edition*** определить:

- конструктор с параметрами типа *string*, *DateTime*, *int* для инициализации соответствующих полей класса;
- конструктор без параметров для инициализации по умолчанию;
- свойства с методами *get* и *set* для доступа к полям типа;
- виртуальный метод *object DeepCopy();*
- свойство типа *int* с методами *get* и *set* для доступа к полю с тиражом издания; в методе *set* свойства бросить исключение, если присваиваемое значение отрицательно. При создании объекта-исключения

использовать один из определенных в библиотеке CLR классов-исключений, инициализировать объект-исключение с помощью конструктора с параметром типа *string*, в сообщении передать информацию о допустимых значениях свойства.

В классе ***Edition*** переопределить (override):

- виртуальный метод *virtial bool Equals (object obj)* и определить операции == и != так, чтобы равенство объектов типа Edition трактовалось как совпадение всех данных объектов, а не ссылок на объекты Edition;
- виртуальный метод int GetHashCode();
- перегруженную версию виртуального метода string ToString() для формирования строки со значениями всех полей класса.

В класс ***Article*** добавить реализации интерфейсов

- *System.IComparable* для сравнения объектов типа *Article* по названию статьи;
- *System.Collections.Generic.IComparer<Article>* для сравнения объектов типа *Article* по фамилии автора.

Определить вспомогательный класс, реализующий интерфейс *System.Collections.Generic.IComparer<Article>*, который можно использовать для сравнения объектов типа *Article* по рейтингу статьи.

В классе ***Magazine*** использовать типы

- *System.Collections.Generic.List<Person>* для списка редакторов журнала;
- *System.Collections.Generic.List<Article>* для списка статей в журнале.

В классе Magazine определить методы для сортировки списка статей

- по названию статьи;
- по фамилии автора;
- по рейтингу статьи.

Определить универсальный делегат

delegate TKey KeySelector<TKey>(Magazine mg);

Определить универсальный класс ***MagazineCollection<TKey>***, содержащий коллекцию объектов типа *Magazine*, в котором для хранения коллекции используется тип *System.Collections.Generic.Dictionary<TKey, Magazine>*. Типовой параметр *TKey* универсального класса *MagazineCollection<TKey>* определяет тип ключа в коллекции *Dictionary<TKey, Magazine>*.

Метод, который используется для вычисления ключа при добавлении элемента *Magazine* в коллекцию класса *MagazineCollection<TKey>*, отвечает делегату *KeySelector<TKey>* и передается *MagazineCollection<TKey>* через параметр единственного конструктора класса.

Класс ***MagazineCollection<TKey>*** содержит

- закрытое поле типа *System.Collections.Generic.Dictionary<TKey, Magazine>*;
- закрытое поле типа *KeySelector<TKey>* для хранения экземпляра делегата с методом, вычисляющим ключ для объекта *Magazine*;
- конструктор с одним параметром типа *KeySelector<TKey>*;
- метод *void AddDefaults()*, с помощью которого можно добавить некоторое число элементов типа *Magazine* для инициализации коллекции по умолчанию;
- метод *void AddMagazines (params Magazine[])* для добавления элементов в коллекцию *Dictionary<TKey, Magazine>*;
- перегруженную версию виртуального метода *string ToString()* для формирования строки, содержащей информацию обо всех элементах коллекции *Dictionary<TKey, Magazine>*, в том числе значения всех полей, включая список редакторов издания и список статей в журнале для каждого элемента *Magazine*;
- метод *string ToShortString()*, который формирует строку с информацией обо всех элементах коллекции *Dictionary<TKey, Magazine>*, содержащую значения всех полей, значение среднего рейтинга статей, число

редакторов издания и число статей в журнале для каждого элемента *Magazine*, но без списков редакторов и статей.

В классе ***MagazineCollection<TKey>*** определить свойства и методы, выполняющие операции со словарем *Dictionary<TKey, Magazine>* с использованием методов расширения класса *System.Linq.Enumerable* и статические методы-селекторы, которые необходимы для выполнения соответствующих операций с коллекцией:

- свойство типа *double* (только с методом *get*), возвращающее максимальное значение среднего рейтинга статей для элементов коллекции; если в коллекции нет элементов, свойство возвращает некоторое значение по умолчанию; для поиска максимального значения среднего рейтинга статей надо использовать метод *Max* класса *System.Linq.Enumerable*;
- метод *Enumerable<KeyValuePair<TKey,Magazine>> FrequencyGroup* (*Frequency value*), возвращающий подмножество элементов коллекции *Dictionary<TKey,Magazine>* с заданной периодичностью выхода журнала; для формирования подмножества использовать метод *Where* класса *System.Linq.Enumerable*;
- свойство типа *IEnumerable<IGrouping<Frequency, KeyValuePair<TKey,Magazine >>>* (только с методом *get*), выполняющее группировку элементов коллекции *Dictionary<TKey, Magazine>* в зависимости от периодичности выхода журнала с помощью метода *Group* класса *System.Linq.Enumerable*.

В методе ***Main***()

1. Создать объект *Magazine* и вызвать методы, выполняющие сортировку списка *List<Article>* статей в журнале по разным критериям, после каждой сортировки вывести данные объекта. Выполнить сортировку

- по названию статьи;
- по фамилии автора;
- по рейтингу статьи.

2. Создать объект *MagazineCollection<string>*. Добавить в коллекцию несколько разных элементов типа *Magazine* и вывести объект *MagazineCollection<string>*.

3. Вызвать методы класса *MagazineCollection<string>*, выполняющие операции с коллекцией-словарем *Dictionary<TKey, Magazine>*, и после каждой операции вывести результат операции. Выполнить

- вычисление максимального значения среднего рейтинга статей для элементов коллекции;
- вызвать метод *FrequencyGroup* для выбора журналов с заданной периодичностью выхода;
- вызвать свойство класса, выполняющее группировку элементов коллекции по периодичности выхода; вывести все группы элементов.

4. Создать объект типа *TestCollection<Edition, Magazine>*. Ввести число элементов в коллекциях и вызвать метод для поиска первого, центрального, последнего и элемента, не входящего в коллекции. Вывести значения времени поиска для всех четырех случаев.

Контрольные вопросы

1. Что представляет собой коллекция (контейнер)? В какой сборке определены коллекции?
2. Что подразумевается под управлением коллекцией?
3. На какие группы можно разделить контейнеры? Какие элементы входят в эти группы?
4. Назовите и охарактеризуйте основные последовательные контейнеры.
5. Что представляют собой ассоциативные множества (словари)? Из каких типов они состоят?
6. Какой класс используется для реализации словарей? Назовите требования к реализации словарей.
7. Приведите синтаксис инициализации словаря и добавления в него элемента.
8. Какие существуют способы проверки отсутствия в словаре ключа?
9. Назовите особенности и приведите примеры инициализации коллекций.
10. Что представляет собой итератор и какие основные возможности он предоставляет?